09

Cliente del servicio

Transcripción

[00:00] Hola a todos y todas, bienvenidos y bienvenidas una vez más a este curso de Servlets. En el video pasado les dije que íbamos a ver cómo hacer para que nuestro servidor consiga devolver un XML si le piden un XML, un JSON si les pide un JSON por ejemplo Angular podría caer un JSON y nuestro sistema Android podría querer un XML. Normalmente los dos quieren JSON, pero bueno, vamos a suponer eso.

[00:26] Entonces para no estar dependiendo de un sistema complejo como Angular y otro como un dispositivo Android, nosotros vamos a crear un cliente que sea capaz de pedir esas dos cosas. Vamos a crear un cliente Java, un proyecto chico en Java solo para poder hacer esos tests. Entonces vamos a hacer eso.

[00:47] Vamos a ir dentro de, voy a cerrar acá esa empresa service y dentro de nuestro proyecto Explorer, en una parte blanca vamos a hacer clic derecho en new other. Vamos a ir a over y acá vamos a hacer clic en ese Java Project. Bien, acá en esta parte nosotros vamos a colocar el nombre del proyecto, vamos a colocar cliente-webservice, vamos a colocarlo así.

[01:18] Y acá, en esa parte de módulos, tienen que destildar esa opción. No es una cosa que vamos a necesitar, finish. Bien, acá nos está preguntando si queremos cambiar de perspectiva, vamos a colocar que no, ya que estamos acostumbrados con esta perspectiva y tenemos acá nuestro proyecto. Vean que es un proyecto Java, tiene la J ahí y no tiene nada dentro. ¿Qué es lo que vamos a necesitar?

[01:47] Para poder hacer una request un usando HTTP en un proyecto Java, nosotros podríamos simplemente ir creando clases y crear todo el Protocolo HTTP, pero vamos a utilizar una biblioteca que nos va a facilitar mucho la vida y vamos a ver que con pocas líneas nosotros vamos a poder hacer una request, un post solamente utilizando esa biblioteca. La biblioteca en sí se llama httpclient, es de Apache, entonces agréguenle a Apache porque hay varias bibliotecas.

[02:23] Entonces, en este primer link nosotros tenemos acá, esta es la página, es dentro del grupo de HttpComponents, y en este caso yo estoy utilizando la versión 5.1 de HttpClient. Yo ya les voy a dejar las bibliotecas ahí listas para que ustedes la bajen así utilizan la misma versión que yo estoy utilizando para no tener problemas.

[02:46] Si nos vamos a un quick start acá vamos a ver cómo configurar esa biblioteca, pero van a ver que es bien simple. Entonces yo ya tengo acá mi biblioteca de HttpClient, vamos a hacer clic derecho, extraer aquí, bien y tengo este libs, acá adentro tengo todos estos .jar que vamos a necesitar, también tenemos el core de httpclient o cliente.

[03:12] Tenemos un o cliente, tenemos un fluent que nos va a servir para poder hacer las requests, y un commons-loggin y un slf4k, que son dependencias que necesita nuestro cliente. Entonces lo que vamos a hacer es lo siguiente. Vamos a crear acá una carpeta dentro de nuestro cliente webservice, clic derecho new folder.

[03:37] Vamos a llamarlo de lib, voy a dejarlo lib, finish. Tenemos esa carpeta y lo que tenemos que hacer ahora es arrastrar los archivos todas esas librerías bien y los llevamos a lib. Copiamos y vamos a tener acá nuestras librerías. Ahora estas librerías por sí solas, si nosotros creamos una carpeta y colocamos las librerías, el proyecto no los toma como que son realmente librerías que tiene que estar mirando.

[04:06] Entonces nosotros tenemos que agregarlos a lo que se llama el path. Vamos a seleccionar todos ellos, todas esas librerías, clic derecho, build path, add to build path. Ahí vemos que aparece ese referenced libraries, librerías, entonces ahí ya está reconociendo esas librerías. Bien.

[04:25] Una vez que agregamos las librerías, vamos a crear una clase que va a ser la que va a hacer esa llamada. Para crear esa clase vamos a hacer clic derecho en src, new class. El paquete se va a llamar com.alura.cliente y el nombre va a ser ClienteWebService. Vamos a agregar ese public void main, para que iniciemos nuestro cliente desde ahí. Y perfecto, tenemos nuestro main, acá.

[05:05] Ahora. ¿Cómo utilizamos ese HttpClient? Para eso vamos a irnos a ese quick start que yo les había dicho, que es esa parte que está acá, a la izquierda. Tenemos una versión larga, más completa, pero para nosotros nos va a servir simplemente esta parte del código. Con esto nosotros ya vamos a poder hacer una request con un post, entonces vamos a hacer un "Ctrl + C" a esa parte. Vamos a llevarlo a nuestro main.

[05:35] "Ctrl + V". Y vamos a tener que prestar atención a algunas cosas, tenemos que importar este request, fíjense, de usar ese que dice fluent. Ese es el request que vamos a utilizar porque hay otros request, entonces tiene que seleccionar esa. Acá este post está dando un error porque está con P mayúscula. Vamos a cambiarlo con p minúscula.

[06:02] Y la página a la que tenía que apuntar no es esa, sino que nosotros tenemos que apuntar a localhost:8080. Tengo todo en mayúscula. Ahora sí. “//localhost:8080/gerenciador/empresas”. Esa es la URL que vamos a hacer el post. Vamos a eliminar este form. No vamos a enviar un formulario ni nada de eso.

[06:31] Y después va a ejecutarse y vamos a esperar la respuesta. Voy a dar un poco de formato a esto acá. Y nos está diciendo que no hemos habilitado se

throw exception. Entonces nos deja acá seleccionar, voy a dejar exception, que es la más genérica. Enter. Y además de esto con, digamos, con esto yo voy a dar un poco de formato, ahí está.

[06:56] Con este request nosotros vamos a hacer un post, ejecutamos y vamos a esperar la respuesta. ¿Qué es lo que nos va a devolver? Nos van a devolver o un JSON o un XML, pero básicamente va a ser un string largo. Entonces vamos a colocar acá string contenido = request. Eso. Ah, ¿qué es lo que ocurre? Es return content nos va a devolver con algún tipo de dato en realidad, JSON, y lo que voy a hacer es colocar acá un .toString.

[07:32] Bien. Ahora sí ese JSON va a ser convertido a string y nosotros lo vamos a dejar ahí en contenido. Fuera de eso, nosotros vamos a querer mostrar ese contenido acá en la consola. Para mostrarlo, nosotros ya lo conocemos, sysout "Ctrl + Espacio" y acá voy a colocar contenido, enter. Perfecto. Así solo más o menos debería funcionar. ¿Ahora qué es lo que ocurre?

[07:59] Acá no estamos diciéndole cuál tipo de datos nosotros queremos obtener, no le estamos diciendo si queremos JSON o XML, entonces vamos a hacer eso. Antes de ejecutar la request y esperar el contenido, vamos a agregar un header, vamos a agregar un encabezado a nuestra request http, y ese header va a tener dos parámetros. Esperen, que ahí hice un clic en un header que no era. "Ctrl + Espacio", vamos a ver que tenemos dos headers.

[08:34] En este caso yo voy a querer el segundo. Este header, nosotros vamos a tener que agregar una palabra llamada accept, que es las personas que crearon el protocolo, dijeron: “Bueno, vamos a colocar la palabra accept, para decir cuáles son los contenidos que nosotros vamos a aceptar de parte de ese servidor”. Entonces colocamos accept y el valor tiene este formato “Application/json” por ejemplo.

[09:07] Entonces le estamos diciendo acá que los valores que nosotros queremos recibir sean del tipo JSON. Esto ahora el servidor lo voy a tener que

comprender. Pero bueno, con esto ya hemos más o menos finalizado la parte de nuestra aplicación web, que va a pedir los datos a nuestro servidor. Ahora lo que tenemos que hacer es que nuestro servidor comprenda esa petición.

[09:32] Voy a hacer un "Ctrl + Shift + O" que tengo unos imports ahí que no estaban siendo usados y ahora sí vamos a ir a nuestro gerenciador, src/main acá de nuestro Servlet y nuestro empresasService. Acá lo que nosotros tenemos que hacer es obtener ese header, ese encabezado. ¿Ese encabezado de dónde lo obtenemos? De la request. Nos la envían por la request.

[09:58] Entonces nosotros tenemos un request.getHeader. Y acá le ponemos el nombre de accept. Ese “Accept” que habíamos puesto acá ese mismo, voy a dar un enter acá, es el mismo que vamos a pedir acá. Y ese request nos está dando un string. Vamos a llamarlo de valor = eso. Ahora en base a ese “Accept” que nos va a devolver ese Application/json o Application/xml si pedimos XML, perdón, apreté mal.

[10:39] Nosotros vamos a tener eso en el valor, entonces ahora tenemos que hacer un simple if. If(valor.) Y acá vamos a ver un truco. Fíjense que ese string finaliza con o xml o json. Los últimos caracteres nos están diciendo cuál es el tipo de dato que nosotros queremos, entonces voy a colocar en endsWith.

[11:07] Y acá le vamos a decir si finaliza en xml, entonces significa que la persona quiere un xml. Entonces vamos a abrir y cerrar llaves, vamos a cortar el código que ya teníamos acá del xml y lo colocamos acá adentro. Voy a ver un poco de formato. Bien, tenemos ahora si es xml, nos envía el xml, else if, si (valor.endsWhth(“json”)) si termina en json, entonces vamos a devolver este código que teníamos acá comentado.

[11:45] Voy a copiarlo acá todo esto así. Voy a descomentar el código "Ctrl + Shift + C". Y vamos a colocar acá, voy a dar un poco de formato. ¿Qué es lo que ocurre si nuestro cliente no pide ni XML ni JSON, un formato que nosotros no conocemos? Bueno, nosotros tenemos que estar preparados para eso, entonces

vamos a hacer un else y vamos a generar un JSON, vamos a generar un archivo JSON. "Ctrl + C" a esta última parte.

[12:18] Pero en vez de enviar un JSON que he creado automáticamente por la biblioteca JSON, nosotros vamos a construir a mano para decirle que no existe un contenido de ese tipo que está pidiendo la persona, entonces vamos a construir a mano un JSON, vamos a abrir llaves, comillas simples y vamos a colocar 'messaje'.

[12:42] Fuera de la comillas simples, dos puntos, abrimos comillas simples de nuevo y colocamos 'No content'. Esto es una forma que es normal, digamos de enviar en caso de que ocurra este escenario, en el que no es ni JSON ni XML que se quiere recibir los datos. Entonces con esto, en teoría tendría que estar funcionando. Vamos a probarlo, guardé, voy a reiniciar el servidor. El cliente, faltó guardar.

[13:13] Voy a pedir como JSON. Y vamos a ejecutar el cliente. Dentro de la función main vamos a hacer un clic derecho en cualquier lugar, run as Java application. Perfecto. Se nos abrió la consola y estamos recibiendo un JSON. Vamos a probar qué pasa si colocamos XML, clic derecho run as Java application.

[13:39] Tengo que guardarlo, lo guardé y recibimos un XML. Bien, ahora vamos a ir al navegador y vamos, ya lo tengo acá, vamos a ir a empresas, enter y recibí un 'message': 'No content', porque no estamos enviándole en realidad ningún formato en particular. En realidad sí, vamos a ver lo siguiente, apretando F12, si yo repito la consulta con un F5, hacemos clic en la request y vemos acá que es dentro de nuestro request header nosotros tenemos ese application/xml.

[14:14] Entonces es uno de los formatos que nosotros aceptamos. ¿Pero qué es lo que ocurre? Nosotros en nuestro código de empresaService colocamos que tiene que finalizar con xml y finalizar con json, cualquiera de esos dos, pero acá no está finalizando con xml. Lo último que finaliza es ese parámetro acá,

entonces lo correcto sería que nosotros viéramos todo este el valor de accept y dividiéramos en cada una de las posibles opciones que nos está pidiendo ese text/html.

[14:55] Nosotros dividiríamos por la coma. Tendríamos cada uno de esos datos y veríamos cuál de ellos nosotros podemos enviar, pero para solucionar eso de forma rápida ahora, lo que podemos hacer es enviar en vez de colocar endsWith podemos colocar contains. Acá también contains en el json, "Ctrl + S", y con eso vamos a resolver ese problema.

[15:19] Vamos a ejecutar el servidor de nuevo. Y vamos a ejecutar nuestro, en este caso xml, vamos a dar un run as java application. Nos está devolviendo xml. Si a ustedes no se les abre la consola automáticamente, me olvidé de decir, tienen este botón acá que es igual que el de la consola, ese botón hace un switch entre las diferentes consolas, entre nuestro cliente y nuestro servlet, nuestra web, nuestra aplicación.

[15:50] Y pueden hacer clic en la flechita para ver cuál de los dos ustedes quieren ver también. Bien. XML funcionó. JSON "Ctrl + S", vamos a ejecuta, run as Java application, devolvió a JSON y ahora acá debería devolver en XML. Vamos a ver. Perfecto. Devolvió en XML. Entonces estamos viendo que esto está funcionando.

[16:16] Lo correcto sería crear una documentación que nuestro gerenciador tuviera una documentación, en un archivo tipo markdown, o sea .md o un txt cualquiera. La idea es que sería bueno tener una documentación que nos diga nuestro gerenciador acepta los tipos XML y JSON y se realiza así la consulta. Entonces sería una cosa buena para mejorar la calidad de nuestro proyecto.

[16:46] Pero por ahora lo vamos a dejar así. Bien, entonces ya nos vemos en la próxima aula para ver cómo hacer un deploy.